UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/770,769	01/25/2001	Avishai Keren	14531.107.1.1	7768

22971 7590 12/18/2006

MICROSOFT CORPORATION
ATTN: PATENT GROUP DOCKETING DEPARTMENT
ONE MICROSOFT WAY
REDMOND, WA 98052-6399

EXAMINER	
VAN HANDEL, MICHAEL P	
ART UNIT	PAPER NUMBER
2623	

DATE MAILED: 12/18/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)
	09/770,769	KEREN ET AL.
	Examiner	Art Unit
	Michael Van Handel	2623

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *01 November 2006*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *17-21,23 and 45-47* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *17-21,23 and 45-47* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)

2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *11/01/2006*.

4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .

5) ☐ Notice of Informal Patent Application

6) ☐ Other: _____.

DETAILED ACTION

Continued Examination Under 37 CFR 1.114

1. A request for continued examination under 37 CFR 1.114, including the fee set forth in 37 CFR 1.17(e), was filed in this application after final rejection. Since this application is eligible for continued examination under 37 CFR 1.114, and the fee set forth in 37 CFR 1.17(e) has been timely paid, the finality of the previous Office action has been withdrawn pursuant to 37 CFR 1.114. Applicant's submission filed on 11/01/2006 has been entered.

Response to Amendment

1. This action is responsive to an Amendment filed 11/01/2006. Claims **17-21**, **23**, **45-47** are pending. Claims **17**, **20**, **45**, **46**, **47** are amended. Claims **1-16**, **22**, **24-44** are canceled. The examiner hereby withdraws the objections to claims **46** and **47** in light of the amendment.

Response to Arguments

1. Applicant's arguments regarding claims **17**, **20**, and **45**, filed 11/01/2006, have been fully considered, but they are not persuasive.

Regarding claims **17**, **20**, and **45**, the applicant argues that the combination of Williams and Richardson does not disclose or suggest an analysis module that compares the original display elements (graphical primitives) with a set of predefined display elements (graphical primitives) and selects modified display elements which are closest to the original display elements to simplify compression in accordance with identified transmission bandwidth

limitations. The examiner respectfully disagrees. As mentioned in the Office Action mailed 6/01/2006, Richardson discloses a technique for updating a framebuffer. The endpoint where changes to the framebuffer originate is known as the VNC server. Changing from one framebuffer state to another is referred to as an update. These updates are created by and sent from the server to the client. The pixel data of the update is divided up into a series of rectangles. Each of these rectangles may be encoded according to a different scheme, dependent on the particular screen content being transmitted and the available network bandwidth. In the example of copy-rectangle encoding, the server identifies the x, y coordinate position of the previous framebuffer state from which the client can copy a rectangle of pixel data. Thus, the encoding on the wire is simply an x, y coordinate. This is useful, for example, when a user moves a window across a screen. Since a previous framebuffer state contains graphical data (a window prior to be moved, for example), which may be re-used as graphical data (a window after being moved, for example) in the new framebuffer update, the examiner maintains that Richardson meets the limitation of comparing "the original display elements (graphical primitives) with a set of predefined display elements (graphical primitives)," as currently claimed. Furthermore, the examiner interprets a screen or rectangle of pixel data to be a graphical primitive, as claimed. Since the graphical data to be re-used in the old framebuffer may be altered (different x-y position, for example) from the graphical data that is needed in the framebuffer update, the examiner maintains that Richardson meets the limitation of selecting "modified display elements which are closest to the original display elements to simplify compression in accordance with identified transmission bandwidth limitations," as currently claimed. This copy-rectangle encoding may comprise only one of the rectangles in the set of

rectangles comprising the update. All screen changes since the last request (described within encoded rectangles) are coalesced into a single update. The update is then sent from the server to the client. Thus, in response to observed screen changes between a previous framebuffer state and a current state, the server creates a set of variously encoded rectangles (the set describing changes between the states) for effectively transmitting an update based on the particular screen content of the update.

Claim Rejections - 35 USC § 103

1. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

2. Claims **17-21**, **23**, **45-47** are rejected under 35 U.S.C. 103(a) as being unpatentable over Williams, Jr. (US 6,202,211) in view of Richardson et al.

Referring to claims **17**, **20**, and **45**, the claimed "remote computing server system that includes a server that provides remote client access to one or more programs that are run at the server, remotely from one or more client systems, and wherein the server converts display commands generated from the one or more programs into compressed video streams" is met as follows:

- The claimed "server, executing a plurality of programs, each of which generates a set of display commands which represent original display elements of a user interface for each of said plurality of programs" is met by Williams, wherein he

teaches a server, which maintains multiple active desktops and applications for display at remotely located STB/TV combinations [col. 3, lines 31-46].

- The claimed "identifying limitations of the client including a compression required by the client, display characteristics of the remote client device, or both" is not expressly disclosed by Williams; however, the Richardson reference teaches using different encoding techniques dependent on the capabilities of the client and the connection between the server and the client. The examiner further notes that the client can request to not be sent data encoded in copy-rectangle encoding, because the client cannot easily read back from its framebuffer. It would have been obvious to one of ordinary skill in the art at the time that the invention was made to modify Williams to include using different encoding techniques dependent on the capabilities of the client, such as that taught by Richardson in order to allow for more efficient bandwidth usage.
- The claimed "analysis module for comparing the original display elements with a set of predefined display elements stored at the server, wherein responsive to transmission bandwidth limitations that are identified by the server, the analysis module selects corresponding modified display elements from the set of predefined display elements that are most similar to one or more of the original display elements, the set of predefined elements compiled to simplify compression in accordance with said transmission bandwidth limitations, <u>wherein the display elements comprise graphical primitives</u>" is not expressly disclosed by Williams; however, the Richardson reference teaches different encoding

techniques, which are used for various video encoding schemes for rendering desktops and other applications generated at a server on a display of a client. Richardson teaches that a connection speed (connection capability) is analyzed and an encoding scheme is chosen based on the capability of the connection from server to client. Changes to the framebuffer originate at the VNC server. When an update is required, the update affects only a small area of the framebuffer. Each rectangle is encoded using a different scheme. The server chooses the encoding most appropriate for the particular screen content being transmitted and the available network bandwidth. All screen changes since the last request are coalesced into a single update. Richardson further teaches copy-rectangle encoding, which copies portions of the video signal instead of using raw data signal, in order to conserve bandwidth. If the client already has the same pixel data elsewhere in its framebuffer, the encoding on the wire is simply an x, y coordinate, which gives the client a position in the framebuffer from which it can copy the rectangle of pixel data. Thus, if a user were to move a window across a screen and an update were requested, the server would choose copy-rectangle encoding for the particular rectangle. It would have been obvious to one of ordinary skill in the art at the time that the invention was made to modify Williams to include encoding different rectangles according to different schemes in response to the particular screen content being transmitted and the available network bandwidth, such as that taught by Richardson in order to save bandwidth.

Note: The language "to simplify compression, utilize a pre-compressed display element, or both" is an advantage of the limitation. It is not a further limitation of the claim.

- The claimed "video compressor which receives the modified display elements selected above and generates there from a compressed video stream for each one of the plurality of programs" is not specifically disclosed in Williams, though the fact that the video information is multiplexed for delivery [col. 7, lines 13-19] would lead one to incorporate the compression teachings of the Richardson document. Richardson discloses Virtual Network Computing, which transmits compressed video images to a client. The compression is discussed with regards to the MPEG standard [page 35, **A Single Graphics Primitive**] for compressing and encoding before transmission. It would have been obvious to one of ordinary skill in the art at the time of the invention to utilize a compressor to compress the video streams before transmission to the client, in order to allow for more efficient bandwidth usage, while, at the same time, complying with compression standards for transmission.
- The claimed "transmitter for the transmission of the plurality of compressed video streams to one or more remote locations" is not expressly disclosed in Williams, though the fact that the video information is multiplexed for delivery [col. 7, lines 13-19] would lead one to incorporate the compression teachings of the Richardson document. Richardson discloses Virtual Network Computing, which transmits compressed video images to a client. The compression is discussed

with regards to the MPEG standard [page 35, **A Single Graphics Primitive**] for compressing and encoding before transmission. It would have been obvious to one of ordinary skill in the art at the time of the invention to transmit compressed video streams to the client, in order to allow for more efficient bandwidth usage, while, at the same time, complying with compression standards for transmission.

Referring to claims **18**, **21**, and **46**, the claimed "set of display elements stored include one or more backgrounds, icons, buttons, menus, or fonts" is not specifically disclosed by Williams; however, Richardson discloses an encoding scheme that takes advantage of the fact that a typical desktop has large areas of solid color and text. The encoding scheme describes rectangles consisting of one majority (background) color and "sub-rectangles" of different colors. Richardson further discloses a pixel-data caching scheme that could efficiently encode multiple occurrences of the same text character by referring to the first occurrence. It would have been obvious to one of ordinary skill in the art at the time that the invention was made to modify Williams to include an encoding scheme that takes advantage of the fact that a desktop has large areas of solid color and text, such as that taught by Richardson in order to allow for more efficient bandwidth usage.

Note: The USPTO considers the applicant's "one or more" language to be anticipated by any reference containing any of the subsequent corresponding elements.

Referring to claims **19**, **23**, and **47**, the claimed "the set of predefined display elements stored differ from the original display elements by one or more color, spatial frequency spectrum, size, contrast, or type" is not specifically disclosed by Williams; however Richardson discloses describing rectangles consisting of a majority (background) color and "sub-rectangles"

of different colors as an effective encoding scheme for taking advantage of the fact that a typical desktop has large areas of solid color. It would have been obvious to one of ordinary skill in the art at the time that the invention was made to modify Williams to include an encoding scheme that takes advantage of the fact that a desktop has large areas of solid color and text, such as that taught by Richardson in order to allow for more efficient bandwidth usage.

Note: The USPTO considers the applicant's "one or more" language to be anticipated by any reference containing any of the subsequent corresponding elements.

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Michael Van Handel whose telephone number is 571-272-5968. The examiner can normally be reached on 8:00am-5:30pm Mon.-Fri..

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Chris Grant can be reached on 571-272-7294. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

CHRIS KELLEY
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2600

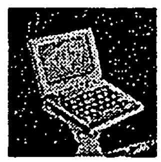
Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

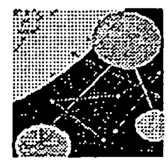
MVH

VIRTUAL NETWORK COMPUTING

Tristan Richardson, Quentin Stafford-Fraser, Kenneth R. Wood, and Andy Hopper*
The Olivetti & Oracle Research Laboratory

VNC is an ultra-thin client system based on a simple display protocol that is platform-independent. It achieves mobile computing without requiring the user to carry any hardware.





The so-called network computer (NC) aims to give users access to centralized resources from simple, inexpensive devices. These devices act as clients to more powerful server machines that are connected to the network and provide applications, data, and storage for a user's preferences and personal customizations. We have taken this idea a stage further. In the virtual network computing (VNC) system, server machines supply not only applications and data but also an entire desktop environment that can be accessed from any Internet-connected machine using a simple *software* NC. Whenever and wherever a VNC desktop is accessed, its state and configuration (right down to the position of the cursor) are exactly the same as when it was last accessed.

In contrast to many recent Internet applications, which have focused on giving users access to resources located anywhere in the world from their home computing environments, VNC provides access to home computing environments from anywhere in the world. Members of the Olivetti & Oracle Research Laboratory (ORL) use VNC to access their personal Unix and PC desktops from any office in our Cambridge building and from around the world on whatever computing infrastructure happens to be available—including, for example, public Web-browsing terminals in airports. VNC thus provides mobile computing without requiring the user to carry any device whatsoever. In addition, VNC allows a single desktop to be accessed from several places simultaneously, thus supporting appli-

*Andy Hopper is also affiliated with Cambridge University Engineering Department.

1089-7801/98/$10.00 ©1998 IEEE

cation sharing in the style of computer-supported cooperative work (CSCW).

The technology underlying VNC is a simple remote-display protocol. It is the simplicity of this protocol that makes VNC so powerful. Unlike other remote display protocols such as the **X Window System** and **Citrix's ICA**, the VNC protocol is totally independent of operating system, windowing system, and applications (see the sidebar, "Thin Clients"). The **VNC system** is freely available for download from the ORL Web site at http://www.orl.co.uk/vnc/.

We begin this article by summarizing the evolution of VNC from our work on thin-client architectures. We then describe the structure of the VNC protocol, and conclude by discussing the ways we use VNC technology now and how it may evolve further as new clients and servers are developed.

THE ORIGINS OF VNC

The X Window System allows applications to display a user interface on a remote machine. ORL extended this functionality in our **Teleporting System** by allowing the user interface of a running X application to be dynamically redirected to a different display.[1,2] Teleporting has been in daily use at ORL for several years now. There are, however, several problems with X that restrict its use in the wide area and, in turn, restrict systems based on it, such as Teleporting:

- X requires the display machine to run an X server program. This heavyweight piece of software requires substantial resources, which machines such as NCs and personal digital assistants (PDAs) cannot be expected to run.
- The X security model makes it inherently dangerous to allow a remote machine to use your display. Accordingly, most system administrators stop X traffic from passing in or out of their sites.
- Application startup is extremely slow on high-latency links due to the number of round-trips performed by a typical application (though there are special proxies that alleviate this problem, such as Low Bandwidth X [LBX][3]).

In addition to these technical problems, there is also the nontechnical problem that X is not Windows, and the world is becoming increasingly Microsoft-dominated.

Videotile: An Ultra-Thin Client

In 1994, ORL built the **Videotile** as an experiment in ultra-thin-client technology. The Videotile is a display device with an LCD screen, a pen, and an ATM network connection. It was designed to display good-quality video, but we also wanted to use it to interact with applications. As a first experiment toward this end, we treated a remote computer screen as a video source and simply shipped the user interface as raw video onto the tile. This worked surprisingly well, but used a significant amount of bandwidth.

By adding a little more intelligence at the application side, we were able still to treat the user interface as video, but to send only those parts of the screen that changed. This idea developed into the VNC protocol.

Java: Access Through a Browser

When Sun Microsystems released the alpha version of the Java language and the HotJava browser in 1995, we realized we could implement the Videotile mechanism in Java to access applications through a Web browser. The thin-client paradigm made the adaptation to Java very straightforward. We wrote the original Java client in a day and the resulting class file was a mere 6 kilobytes in size. This eventually became the VNC applet described in more detail elsewhere.[4] Any Java-capable browser could now provide access to a user's desktop, giving the mobility of the Teleporting system, but on a global scale.

THIN CLIENTS

The Virtual Networking Computing (VNC) system is a thin-client system. Like all such systems, it reduces the amount of state maintained at the user's terminal. VNC viewers are exceedingly thin because they store no unrecoverable state at the endpoint. This contrasts with systems like X Windows, and allows arbitrary disconnection and reconnection of the client with no effect on the session at the server. Since the client can reconnect at a different location—even on the other side of the planet—VNC achieves mobile computing without requiring the user to carry computing hardware.

Of course, VNC is not the only thin-client system. Others include those built around the Citrix ICA protocol (for example, Citrix's Winframe and Insignia Solutions' Ntrigue), SCO's Tarantella, Graphon's RapidX, and Microsoft's Windows-based Terminal Server (previously code-named Hydra). The problem with all of these systems except Microsoft's is that, unlike X, they use proprietary protocols, so reliable information about them is difficult to obtain. Citrix's ICA protocol is a popular mechanism for remote interaction with PCs, but it appears to be closely tied to the Microsoft Windows GUI, so it may not be an ideal general-purpose remote display protocol.

Microsoft has developed its own protocol, T.Share, based on the ITU T.120 protocol.[1] This is already used in Microsoft's NetMeeting conferencing software product. Preliminary details suggest that Microsoft's protocol is more like VNC than ICA—the Hydra white paper refers to a "super-thin" client.

We hope that VNC, or something like it, can become an open cross-platform standard for very-thin-client computing.

REFERENCE

1. "Microsoft Windows NT 'Hydra' and Windows Based Terminals," white paper, available at http://microsoft.com/ntserver/guide/hydrapapers.asp.

THE VNC PROTOCOL

The technology underlying the VNC system is a simple protocol for remote access to graphical user interfaces. It works at the framebuffer level and therefore applies to all operating systems, windowing systems, and applications—indeed to any device with some form of communications link. The protocol will operate over any reliable transport such as TCP/IP.

The endpoint with which the user interacts (that is, the display and/or input devices) is called the *VNC client* or *viewer*. The endpoint where changes to the framebuffer originate (that is, the windowing system and applications) is known as the *VNC server* (see Figure 1).

VNC is truly a "thin-client" system. Its design makes very few requirements of the client, and therefore simplifies the task of creating clients to run on a wide range of hardware.

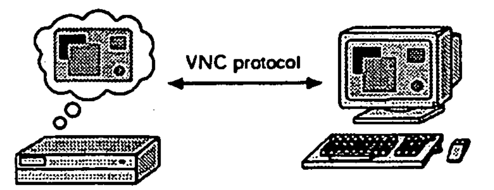


Figure 1. VNC architecture.

A Single Graphics Primitive

The display side of the protocol is based on a single graphics primitive:

Put a rectangle of pixel data at a given *x*, *y* position.

At first glance this might seem an inefficient way to draw some user interface components. However, allowing various encoding schemes for the pixel data gives a large degree of flexibility in trading off parameters such as network bandwidth, client drawing speed, and server processing speed.

The lowest common denominator is the so-called *raw encoding*, where the pixel data for a rectangle is simply sent in left-to-right scanline order. All VNC clients and servers must support this encoding. However, the encodings actually used on a given connection can be negotiated according to the capabilities of the server and client and the connection between them.

For example, *copy-rectangle encoding* is very simple and efficient, and can be used when the client already has the same pixel data elsewhere in its framebuffer. The encoding on the wire is simply an *x*, *y* coordinate. This gives a position in the framebuffer from which the client can copy the rectangle of pixel data. This encoding is typically used when the user moves a window across the screen or scrolls a window's contents.

Most clients will support copy-rectangle encoding, since it is generally easy to implement, saves bandwidth, and is likely to be faster than sending raw data again. However, in a case where a client cannot easily read back from its framebuffer, the client could specify that it should *not* be sent data encoded this way.

A typical workstation desktop has large areas of solid color and text. One of our most effective encodings takes advantage of this phenomenon by describing rectangles consisting of one majority (background) color and "sub-rectangles" of different colors. There are numerous other possible schemes. We could use a JPEG encoding for efficient transmission of still images or an MPEG encoding for moving images. A pixel-data caching scheme could efficiently encode multiple occurrences of the same text character by referring to the first occurrence.

Adaptive Update

A set of rectangles of pixel data makes a *framebuffer update* (or simply, *update*). An update represents a change from one valid framebuffer state to another. In this sense, an update is similar to a frame of video. It differs, however, in that it usually affects only a small area of the framebuffer. Each rectangle may be encoded using a different scheme. The server can therefore choose the encoding most appropriate for the particular screen content being transmitted and the available network bandwidth.

The update protocol is demand-driven by the client. That is, an update is only sent by the server in response to an explicit request from the client. All screen changes since the client's last request are coalesced into a single update. This gives the protocol an adaptive quality: the slower the client and the network, the lower the rate of updates. On a fast network, for example, as the user drags a window across the screen it will move smoothly, being drawn at all the intermediate positions. On a slower link—for example, over a modem—the client will request updates less frequently, and the window will appear at fewer of these positions. This means that the display will reach its final state as quickly as the network bandwidth will allow, thus maximizing the speed of interaction.

Input

The input side of the VNC protocol is based on a standard workstation model of a keyboard and multibutton pointing device. The client sends input events to the server whenever the user presses a key or pointer button, or moves the pointing device. Input events can also be synthesized from other nonstandard I/O devices. On the Videotile, for example, a pen-based handwriting recognition engine generates keyboard events.

Connection Setup and Shutdown

To establish a client-server connection, the server first requests authentication from the client, using a challenge-response

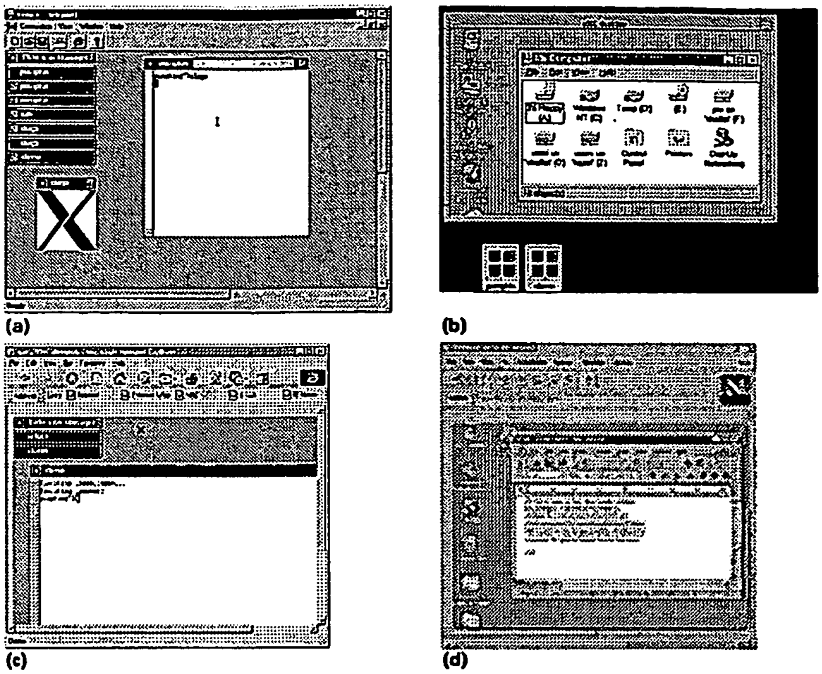


Figure 2. A variety of desktops being accessed from different viewers: (a) a Unix desktop from a Windows viewer, (b) a Windows 95 desktop from an X viewer, (c) a Unix desktop from a Java applet within Internet Explorer, and (d) a Windows desktop using Netscape on Unix.

scheme; the client typically requires the user to enter a password at this point. The server and client then exchange messages to negotiate desktop size, pixel format, and encoding schemes. The client requests an update for the entire screen, and the session begins. Because of the stateless nature of the client, either side can close the connection at any time without adverse consequences.

VNC Viewers

In day-to-day use, we prefer the more descriptive term *viewer* to the rather overloaded word *client*. Writing a VNC viewer is a simple task, as indeed it should be for any thin-client system. It requires only a reliable transport (usually TCP/IP), and a way of displaying pixels (either writing directly to the framebuffer or going through a windowing system).

We have written viewers for all the networked display devices available at ORL. These include the Videotile (the original VNC viewer), an X-based viewer (which runs on Solaris, Linux, and Digital Unix workstations), a Win32 viewer that runs on Windows NT and 95, and a Java applet that runs on any Java-capable browser (including Sun's JavaStation). Members of our lab use these viewers on a daily basis to access their personal computing environments.

The images in Figure 2 show a variety of X and Windows desktops being accessed from both Java and native X and Windows viewers.

VNC Servers

Writing a VNC server is slightly harder than writing a viewer. Because the protocol is designed to make the client as simple as possible, it is usually up to the server to perform any necessary translations (for example, the server must provide pixel data in the format the client wants). We have written servers for our two main platforms, X (that is, Unix) and Windows NT/95.

The X-based server was the first one we developed. A single Unix machine can run a number of VNC servers for different users, each representing a distinct VNC desktop. Each desktop is like a virtual X display, with a root window on which several X applications can appear.

The Windows VNC server was a little more difficult to create. Windows has fewer places to insert hooks into the system to monitor display updates, and the model of multiuser operation is less clearly defined. Our current server simply mirrors the real display to a remote client, which means that only a single VNC desktop is available from any one PC.

The X-based server, the X viewer, the Win32 server, and Win32 viewer can all fit on a single floppy disk.

We have also created "thin" servers which produce displays other than desktops, using a simple toolkit. A "VNC CD player," for example, generates a CD player user interface using VNC directly without any reference to a windowing system or framebuffer (see figure 3 on the following page). Such servers can run on very simple hardware, and can be accessed from any of the standard VNC viewers.

ANY USER INTERFACE, ANYWHERE

At ORL, we have used VNC to add mobility to workstation GUIs, where the concept of at least some form of remote interaction is not new. But the protocol's simplicity could allow it to be used on a much wider range of hardware. Consumer electronics devices, such as CD players, usually have a highly specialized user interface and typically employ customized phys-

ical display devices. This has traditionally prevented such interfaces from being mobile in the VNC sense of the word.

But we think VNC's usefulness can be extended so that users could, for example,

- bring up the controls for their video recorder on a mobile phone as they drive home from work,
- use a modem to dial a telephone answering machine and reprogram it through a graphical interface,
- display their car stereo or GPS receiver as part of the dashboard, regardless of the equipment brand installed.

At present, such functions require the displaying device to have detailed knowledge of the remote system and to emulate that system's user interface or some alternative interface that it deems appropriate. For example, you would need a driver for your video recorder, which was designed for your mobile phone's operating system. A much simpler approach would be to use the interface designed for and provided with the remote device, but to interact with it locally.

For this we need a set of common "phonemes" with which we can construct a variety of GUIs. This is the role that the VNC protocol—or something very similar to it—can play. It is simple enough to implement cheaply in consumer electronics hardware, yet it can be used to describe the building blocks of most modern user interfaces. With standards such as **IEEE-1394 Firewire**, **USB**, and **IrDA**, we have the physical interface to connect a variety of devices; with VNC, we can add a standard for plug-and-play user interfaces. Imagine walking up to any workstation, connecting your PDA to the USB port, and having the PDA applications instantly available on the workstation screen, or plugging your PDA into your car and having the engine management unit display servicing information on the PDA's screen. And imagine that this works for any workstation, any PDA, any car.

The engine management example illustrates an important point: A standardized GUI protocol allows devices that have no physical display of their own to provide graphical information when such a display becomes available to them. Your PDA could, perhaps, shrink to the size of a pen if it could access a display and keyboard through an IrDA link. And yet this "microPDA" could still display PowerPoint-style presentations when in the vicinity of an LCD projection panel or a large TV.

This model is very similar to the Web, where services without an I/O capability of their own wait for a user to provide one in the form of a Web browser. The success of this strategy has led to embedding HTTP daemons in printers, switches, routers, and other devices. But to be a Web server, a device must at least have a TCP stack and an IP address. And to be a Web browser requires at least the ability to render fonts and parse HTML.

In contrast, VNC requires only a reliable transport medium and the simplest of display capabilities. And while a page of HTML will generally require the transmission of fewer bytes than its VNC equivalent, the latter is infinitely more flexible.

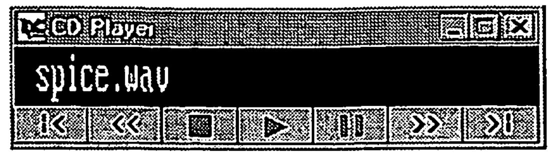


Figure 3. Remote access to a CD player control panel using the VNC system.

FUTURE WORK

We are now building VNC software for a variety of desktop platforms, but it would not be difficult to make remote access practical for a wider range of devices. We can envisage cheap hardware that might, for example, drive a 7-segment LCD and also emit a VNC equivalent over a USB or RS232 link. The VNC commands to draw and erase each segment could be stored as a sequence of bytes in a small amount of ROM and sent over a communications link when the segment is lit or switched off. Hardware such as this, if made in quantity, could be very cheap and could allow for mobility of much more than just a conventional "desktop." If built into television sets, VNC viewers could allow them to act as displays for a very wide range of devices—including, of course, the PC at the office. ■

ACKNOWLEDGMENTS

We would like to thank our colleagues Paul Webster, James Weatherall, and Andy Harter for their contributions to the project and their assistance in creating the VNC system.

URLs for this article

Citrix's ICA • www.citrix.com/technology/icatech.htm
Graphon's RapidX • www.graphon.com/
IEEE-1394 Firewire • firewire.org/
Insignia Solutions' Ntrigue • www.insignia.com/
IrDA • www.irda.org/
Microsoft's Terminal Server (Hydra) • www.microsoft.com/ntserver/info/hydra.htm
ORL's Teleporting System • www.orl.co.uk/teleport/
ORL's Videotile • www.orl.co.uk/tile.html
ORL's VNC system • www.orl.co.uk/vnc/
SCO's Tarantella • www.tarantella.sco.com/
USB • www.usb.org/
X Window System • www.opengroup.org/tech/desktop/x/

REFERENCES

1. T. Richardson et al., "Teleporting in an X Window System Environment," *IEEE Personal Comm.*, No. 3, 1994, pp. 6-12. Also available as ORL Technical Report 94.4, ORL, Cambridge CB2 1QA, England.
2. T. Richardson, "Teleporting—Mobile X Sessions," *Proc. 9th Ann. X Technical Conf.*, Jan. 1995. Also in *The X Resource*, Issue 13, O'Reilly & Associates, Jan. 1995. Also available as ORL Technical Report 95.5, ORL, Cambridge CB2 1QA, England.
3. Open Group, "X11R6.3 (Broadway) Overview," http://www.opengroup.org/tech/desktop/x/broadway.htm#lbx (current September 1997).
4. K.R Wood et al., "Global Teleporting with Java: Toward Ubiquitous Personalized Computing," *Computer*, Vol. 30, No. 2., Feb. 1997, pp. 53-59. Also available as ORL Technical Report 96.2, ORL, Cambridge CB2 1QA, England.

Tristan Richardson is a research scientist at ORL, and his research interests include mobile and network computing, windowing systems, and multimedia. He holds an MA in computer science and an MPhil in computer speech and language processing from the University of Cambridge.

Quentin Stafford-Fraser is a research scientist at ORL, and his chief research interests are personalized mobile computing, in-car information systems, and novel user interfaces. Before joining ORL he worked at Rank Xerox EuroPARC (now XRCE) on video-augmented environments. He holds an MA and a PhD from the University of Cambridge.

Kenneth R. Wood is a research scientist at ORL. His interests include mobile computing, multimedia, concurrency theory, and applied formal methods. Before joining ORL, he worked as a member of the scientific staff at Nortel Technology and taught at Oxford University. He received the AB degree in applied mathematics from Harvard University and the MSc and DPhil degrees in computation from Oxford University.

Andy Hopper is director of the Olivetti & Oracle Research Laboratory (ORL) in Cambridge, a director of Advanced Telecommunications Modules Limited, and chair of Telemedia Systems Ltd. He is also a professor of communications engineering at the University of Cambridge and a fellow of Corpus Christi College. His research interests include networking, multimedia, and mobile systems. He received the BSc degree from the University of Wales in 1974 and the PhD degree from the University of Cambridge in 1978. He is a fellow of the IEE and the Royal Academy of Engineering.

Readers may contact the authors at ORL, 24a Trumpington Street, Cambridge CB2 1QA, UK; email: vnc@orl.co.uk.